Discrete Model Prodective Control on a Bicycle Dynamic Model

Apurva Sontakke,Nalin Bendapudi, Varun Shetty
{apurvaps, bnalin, vashetty}@umich.edu

***Abstract*—This report chronicles our the work done in EECS561-Digital Control Systems Project. The objective was to do trajectory optimization-based control of a car so that it smoothly traverses a given track. We have assumed that the dynamics of the car work according to the bicycle model. A proportional controller was used to obtain the initial trajectory of the car. The dynamic model was linearized around this trajectory, and a model predictive controller to find an optimized trajectory.**

***Index Terms*—Model predictive control, linearization, quadprog, bicycle model.**

I. Introduction

In this section, we'll talk about the dynamic model we used, our problem statement, and the motivation behind using MPC.

A. Problem Statement

Our team proposes to control an autonomous vehicle modeled by the bicycle model. The objective will be to track a pre-defined race-track whose Cartesian coordinates are known. This builds on the controls project of the Self Driving Cars course, which all three of us had taken in Fall 2019. For the 535 project, we had used a PID controller but here we plan on using the discrete time MPC controller.We are planning to generate a trajectory by using discrete time MPC given the initial states such that it lies between the left border and the right border of the track and reaches the specified end position.The lateral tracking error that we hope our controller will handle is +/- 2m.

Figure 1 shows the track we are using.

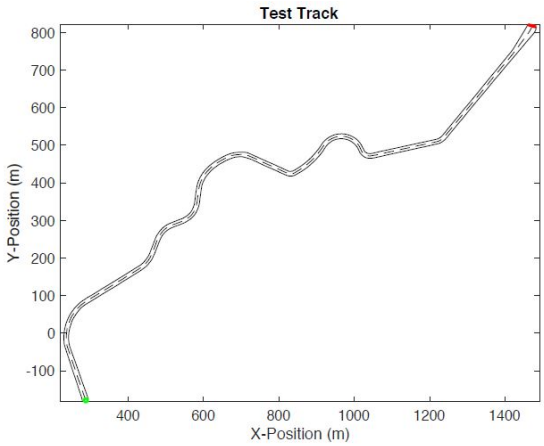

Fig. 1. Original Track

B. Bicycle Model

The non-linear bicycle model is given as follows:

$$\begin{bmatrix} \dot{X} \\ \dot{u} \\ \dot{Y} \\ \dot{v} \\ \dot{\psi} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} ucos\psi - vsin\psi \\ \frac{1}{m}(-fmg + N_w F_x - F_{yf} sin(\delta_f)) + vr \\ usin\psi + vcos\psi \\ \frac{1}{m}(F_{yf} cos(\delta_f) + F_{yr}) - ur \\ r \\ \frac{1}{I_z}(aF_{yf} cos(\delta_f) - bF_{yr}) \end{bmatrix} \tag{1}$$

The lateral forces F_{yf} and F_{yr} are described using Pacejka "Magic Formula"

$$F_{zf} = \frac{b}{a+b} mg \tag{2}$$

$$F_{yf} = F_{zf} D_y sin(C_y \tan^{-1}(B_y \phi_{yf})) + S_{vy} \tag{3}$$

$$F_{zr} = \frac{a}{a+b} mg \tag{4}$$

$$F_{yr} = F_{zr} D_y sin(C_y \tan^{-1}(B_y \phi_{yr})) + S_v y \tag{5}$$

where

$$\phi_{yf} = (1 - E_y)(\alpha_f + S) \tag{6}$$

$$\phi_{yr} = (1 - E_y)(\alpha_r + S_{hy}) + \frac{E_y}{B_y} \tan -1(B_y(\alpha_r + S_{hy})) \tag{7}$$

where α_f and α_r are the front and rear lateral slip angles which are given in degrees in the previous formulas. The front and rear lateral slip angles which is described in radians is given by:

$$\alpha_f = \delta_f - \tan^{-1}(\frac{v + ar}{u}) \tag{8}$$

$$\alpha_r = -\tan^{-1}(\frac{v - br}{u}) \tag{9}$$

Additionally, combined longitudinal and lateral loading of tires will be limited to F_x^* and F_{yr}^* in the following manner:

$$F_{total} = \sqrt{(N_w F_x)^2 + (F_{yr})^2} \tag{10}$$

$$F_{max} = 0.7mg \tag{11}$$

If $F_{total} > F_{max}$:

$$F^x = \frac{F_{max}}{F_{total}} F_x \tag{12}$$

$$F^{yr} = \frac{F_{max}}{F_{total}} F_{yr} \tag{13}$$

The inputs into this model are δ_f , which is the front wheel steering angle; and F_x, which is the traction force generated at each tire by the vehicle's motor. The vehicle begins from the following initial condition:

$$\begin{bmatrix} x \\ u \\ y \\ v \\ \psi \\ r \end{bmatrix} = \begin{bmatrix} 287[m] \\ 5[m/s] \\ -176[m] \\ 0[m/s] \\ 2[rad] \\ 0[rad/s] \end{bmatrix} \tag{14}$$

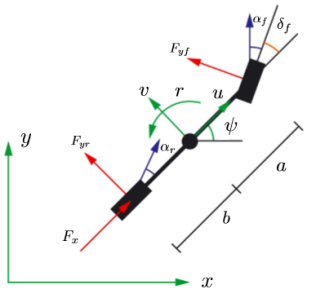

Fig. 2. An illustration of the bicycle model used to define the vehicle's dynamics.

C. Motivation to do MPC

MPC (Model Predictive Control) is a time control strategy in which the future steps i.e the sequence of the control steps is determined my minimizing the cost function (objective function) at each time step over a horizon dependent upon the equations and constraints of the model.In short Model Predictive Control is used to predict the future behaviour of the system. One of the most important advantage of using MPC is that it can handel multiple input and multiple output systems and interactions between them.

We use the linear MPC because of its ability to find a global and optimal minimum by placing constraints in the inputs and states. We use the linear MPC algorithm where we apply successive linearization to develop a time-variant linear system.

We have assumed full state feedback in our control procedure throughout. But this is seldom possible in the real world. Usually we would have to design an observer to get the state estimates. These estimates could be different from our real states. This is where MPC performs well. In our project we tested MPC with an initial state different from that given for the reference trajectory and it still gave good results.

TABLE I
DYNAMIC MODEL CONSTANTS

Vehicle Parameter	Value
δ	[-0.5,0.5]
F_x	[-5000,5000]
m	1400
N_w	2
f	0.01
I_x	2667
a	1.35
b	1.45
B_y	0.27
C_y	1.2
D_y	0.7
E_y	-1.6
S_{hy}	0
S_{vy}	0
g	9.806

II. METHODOLOGY

We obtain a linear model of our dynamics by calculating an error model about the reference trajectory. In order to generate the reference trajectory we used a Proportional controller by calculating the lateral error between the center-line of the track and the current position of our vehicle. This error is used as the feedback reference to then generate the steering command.

A. Trajectory Tracking using Proportional Controller

To achieve an initial estimate of the trajectory that the MPC will have to track, a simple P controller was

used. Cross track error was used as the error term for the controller. Cross track error is the lateral distance between the track's center line and the vehicle's position. This controller was forward simulated using MATLAB's ODE45 with a time step of 0.01s. The resultant states and control inputs were stored and used in the next section.

B. Trajectory Tracking using Discrete MPC

A basic block diagram for MPC is shown below:

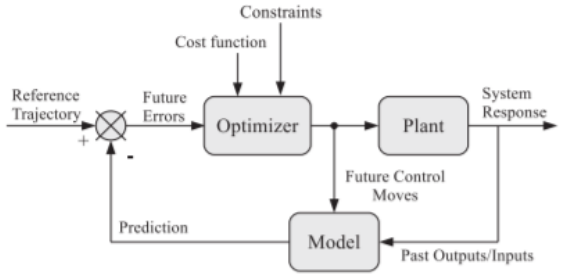

Fig. 3. MPC Block Diagram

1) Modelling: While obtaining the nominal trajectory we set a constant longitudinal velocity and this leads to a reduced order bicycle model. The tire dynamics are assumed to be linear which is a fair assumption for relatively low speeds. This reduced order model is:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{v} \\ \dot{\psi} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} ucos\psi - vsin\psi \\ usin\psi + vcos\psi \\ \frac{1}{m}(F_{yf}cos(\delta_f) + F_{yr}) - ur \\ r \\ \frac{1}{I_z}(aF_{yf}cos(\delta_f) - bF_{yr}) \end{bmatrix} \tag{15}$$

2) Discretization: In order to create a decision vector and formulate the problem such that it can be fed to an optimizer, the model needed to be discretized. We chose apply Euler's integration to generate a discrete time system by approximating the dynamics as follows:

$$x(k+1) = x(k) + dt \times (A(k)x(k) + B(k)u(k)) \tag{16}$$

where $x(k) \in \mathbb{R}^n, A(k) \in \mathbb{R}^{n\times n}, B(k) \in \mathbb{R}^{n\times m}$. We discretized the system at a sampling time of 0.01s and used a prediction horizon of 10 steps. For our model, A(k) and B(k) are as follows:

$$A(k) = \begin{bmatrix} 0 & 0 & -sin(\psi_{ref}) & A_{13} & 0 \\ 0 & 0 & cos(\psi_{ref}) & A_{24} & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & A_{43} & 0 & A_{45} \\ 0 & 0 & A_{53} & 0 & A_{55} \end{bmatrix} \tag{17}$$

where

$$A_{13} = -u_0 sin(\psi_{ref}) - v_{ref} cos(\psi_{ref})$$
$$A_{24} = -v_{ref} sin(\psi_{ref}) + u_0 cos(\psi_{ref})$$
$$A_{43} = -\frac{Ca_r + aCa_f cos(\delta_f)}{mu_0}$$
$$A_{45} = -\frac{Ca_r b + Ca_f a cos(\delta_f) - u_0}{mu_0}$$
$$A_{53} = \frac{Ca_r b - aCa_f cos(\delta_f)}{I_z u_0}$$
$$A_{55} = -\frac{b^2 Ca_r + a^2 Ca_f cos(\delta_f)}{I_z u_0}$$

$$B(k) = \begin{bmatrix} 0 \\ 0 \\ B_3 \\ 0 \\ B_5 \end{bmatrix} \tag{18}$$

where

$$B_3 = \frac{Ca_f(cos(\delta_f) - \delta_f sin(\delta_f) + \frac{v_{ref} + ar_{ref} sin(\delta_f)}{u_0}}{m}$$

$$B_4 = \frac{aCa_f(cos(\delta_f) - \delta_f sin(\delta_f) + (\frac{(v_{ref} + ar_{ref}) sin(\delta_f))}{u_0}}{I_z}$$

These $A(k)$ and $B(k)$ matrices are basically the Jacobians of A and B with respect to the states and input, respectively.

3) MPC implementation: To find the optimized solution we use quadratic programming that is specified by:

$$min \frac{1}{2} x^T H x + f^T x \tag{19}$$

such that $Aeq * x = beq$ and $lb \leqslant x \leqslant ub$. H and Aeq are matrices and beq, lb abd ub are vectors.

The initial conditions of the dynamics are satisfied in the first three rows of Aeq and beq. All subsequent rows of beq are zero. Each subsequent row of Aeq represents the Euler integration steps in order from time step k to time step $k + 10$. Aeq encodes the linear equalities and is a $M \times N$ matrix where M are the number of equalities i.e. in our case 55 (number of states $\times$ number of state decision variables); N is the number of total decision variables. beq is a vector with M rows. An example of Aeq for the first time step is:

$$\begin{bmatrix} Aeq(6:10, 6:10) \\ Aeq(6:10, 1:5) \\ Aeq(6:10, 56) \end{bmatrix} = \begin{bmatrix} I_5 \\ A(2) \\ B(2) \end{bmatrix} \tag{20}$$

The remaining elements of Aeq can be filled in a similar fashion.

H is the quadratic objective term. We construct it using a quadratic penalty of Q and R on state matrix and input

matrix respectively. We have used the following values for Q and R.

H should penalize all the states and inputs, and Q and R do exactly the same. So, we construct H using Q and R such that it is a diagonal matrix with Q and R repeated throughout the prediction horizon penalizing the states and inputs at the corresponding time steps.

$$H = \begin{bmatrix} Q & & R & & \\ & \ddots & & \ddots & \\ & & Q & & R \end{bmatrix}$$

We have used a higher penalty on the heading as compared to the other states because we observed that the car would oscillate without that penalty. The heading term indirectly acts as the derivative of the lateral error and hence was heavily penalized.

We compute the states using `ode45` and using the inputs generated from MPC.

III. Results and Discussion

The initial trajectory (after applying the proportional control) and the final trajectory (after applying MPC) is shown in figures 4 and 5 respectively.

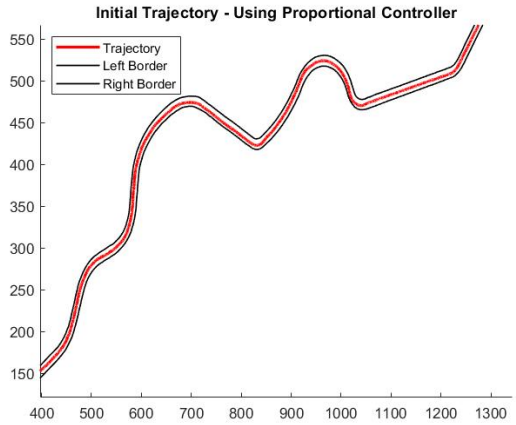

Fig. 4. Initial trajectory generated using Proportional Controller (zoomed in to show the most difficult-to-maneuver parts of the track)

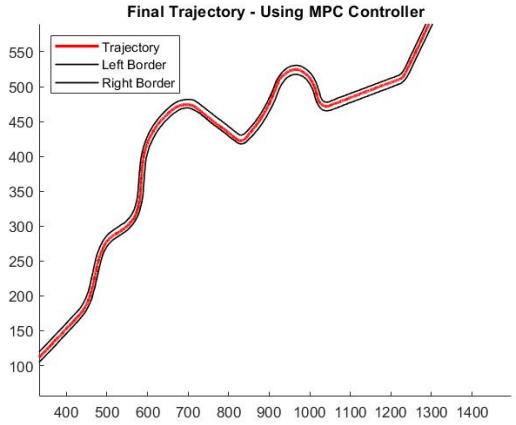

Fig. 5. Final trajectory generated using MPC (zoomed in to show the most difficult-to-maneuver parts of the track)

We observe that the MPC trajectory tracks the reference trajectory perfectly. This is expected since the reference trajectory (or the initial trajectory, as it was called earlier) does a very good job of traversing the track.

Next, we see what happens if the MPC is provided with a different initial state. The new initial state provided is: $[284, 5, -180, 0, 2, 0]$ (each state has the same units as in Eqn. 14) Basically we start from the coordinates $(284, -180)$ instead of $(287, -176)$. The trajectory tracked by MPC is shown in Fig. . We also plot the initial part of the track in Fig. 7 to compare initial and final trajectories when provided with different initial states.

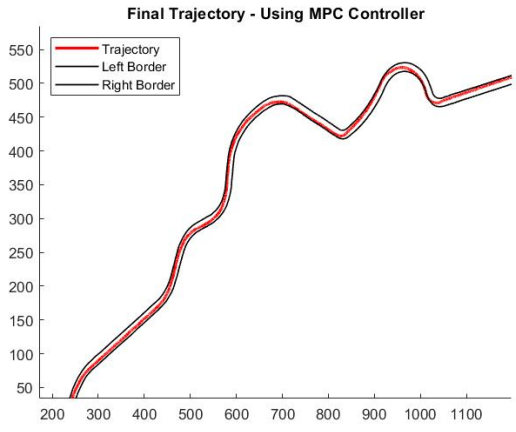

Fig. 6. Final trajectory generated using MPC with a different Initial State (zoomed in to show the most difficult-to-maneuver parts of the track)

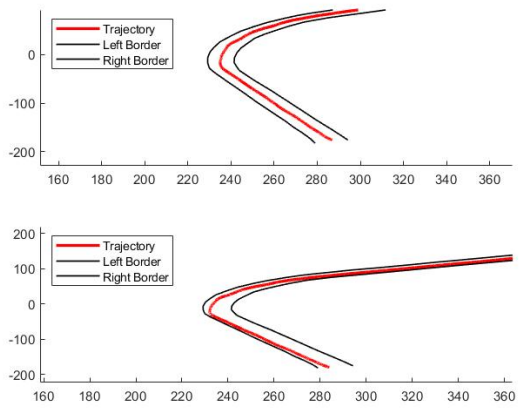

Fig. 7. Comparison of both trajectories in the initial part of the track when provided with different initial states

Finally, we also tried to introduce random noise to the state for each iteration inside MPC's control loop. This is supposed to model the disturbances in state estimation. The final trajectory generated was still acceptably within bounds. The noise is added using MATLAB's `randn` command, and is equal to 5% of the initial state times $\mathcal{N}(0, 1)$. The trajectory is shown in Fig. 8.

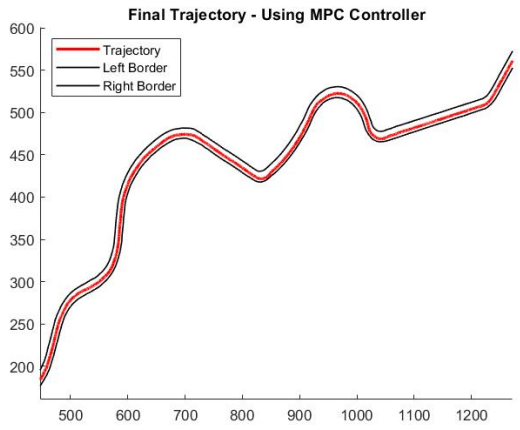

Fig. 8. Final trajectory generated using MPC with noise added to states (zoomed in to show the most difficult-to-maneuver parts of the track)

IV. Conclusion

The objective in the problem statement has been satisfied. The car modeled by the bicycle model tracks a reference trajectory even though the MPC controller used a reduced-order dynamic model, and initial state was altered. The code for the MPC controller is documented and open-sourced on GitHub. We highly recommend that the reader have a look at the code (link provided in the Appendix) and runs the code. This seems to be the only way to reproduce and verify the results shown in this report.

References

[1] F. Kühne, J. M. G. da Silva Jr., and W. F. Lages, "Mobile robot trajectory tracking using model predictive control," *GCAR - Grupo de Controle Automação e Robótica*.

[2] M. Charest, R. Dubay, and S. Everett, "Complex trajectory tracking using mpc with prediction adjustment," *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2013.

[3] Q. Hu, J. Xie, and C. Wang, "Dynamic path planning and trajectory tracking using mpc for satellite with collision avoidance," *ISA Transaction on Science Direct*, vol. 84, pp. 128–141, 2019.

[4] M. F. Li Dai, Yuanqing Xia and M. S. Mahmoud, *Discrete-Time Model Predictive Control, Advances in Discrete Time Systems*. IntechOpen, DOI: 10.5772/51122, 2012, Chap. 4.

V. Contribution

Apurva Sontakke	Code: Problem formulation, Discretization, Simulation and Analysis Report documentation
Nalin Bendapudi	Code: Simulation and Analysis Report documentation
Varun Shetty	Code: Simulation and Analysis Report documentation

VI. Appendix

The code for this project is uploaded on GitHub and can be accessed through this link.